Interactive Program Execution in Lispedit

Martin Mikelsons

Computer Science Department
IBM T. J. Watson Research Center
Yorktown Heights, New York

Abstract: We describe a powerful interactive debugger (Heval) embedded in a larger programming environment (Lispedit). In Lispedit, the programmer creates, edits and files Lisp functions from a uniform editor interface that makes use of the structure of Lisp expressions. With Heval, the programmer can execute Lisp expressions and functions in a very natural way through the editor interface. We describe the debugging interface and how it has evolved over several years of use.

INTRODUCTION

The purpose of debugging tools is to allow a programmer to observe the behavior of a program in order to detect and remove the causes of undesirable behavior. Batch debugging tools produce trace information from which the behavior can be inferred. Interactive tools interrupt or slow down execution so that it can be observed more directly. Both kinds of tools interact with the program at individual points, where

- trace information is generated,
- automatic tracing is initiated or stopped,
- execution is interrupted or the rate is changed.

There is a potential to generate an overwhelming amount of information that can obscure failures and slow down the search for errors. The volume of information can be controlled by careful planning and analysis of the program.

The planning has to be done by the programmer because the terms in which traditional debuggers refer to the program are not related to the units that compose the program. The most primitive tools simply refer to the program in its machine language form and the programmer must map source

statements and expressions to specific machine instructions by examining compiler generated listings. More advanced tools <plic> perform some of this mapping automatically and allow the programmer to refer to source lines or statements.

But high-level programming languages encourage the programer to compose programs by combining and nesting small units into bigger compound units. Thus most of the logical units of a program consists of more than one line or statement. If machine instructions, source lines or statements are the units of program reference, the programmer must identify the larger units by careful analysis of the source listing. For example, to examine the program after a compound conditional statement has been executed, the programmer must examine the source to identify all possible exits and set break-points on them.

We describe here a debugging tool that operates in an environment where the programer refers to a program in terms of the natural units defined in the programming language. Compound units are easily and naturally identified by pointing with an appropriate editor. The programmer never has to count lines or statements in order to identify the extent of a well defined unit of the programming language. We believe that by applying the debugging tool to extended areas of the program, as opposed to individual points, we have created a more natural and powerful debugging interface.

The Lispedit Programming Environment

Lispedit is an integrated programming environment for Lisp programs <apdt>. In Lispedit, the user can creates, edits and files Lisp functions and expressions in an environment that makes use of the structure of these objects.

Lispedit is a structure editor for Lisp s-expressions. Since Lisp programs and their data are both s-expressions, the same interface is used to examine and modify both programs and data.

An important feature of Lispedit is a program display that shows the structure of a Lisp expression from the point of view of a selected sub-expression. The sub-expression, called the focus, is shown highlighted and its relationship to the surrounding context is shown by automatically generated indentation. Selected components of the focus and its context are elided (shown as '...') in order to condense large expressions to the confines of a finite screen.

The Debugging Interface in Lispedit

When we expanded the function of Lispedit to include an execution facility, our goal was to use the editor interface, and the structure of